

# Analyse statique

Rayan Bachir Lachguel

20 mars 2025

## 1 CERT C EXP33-C. Do not read uninitialized memory

### 1.1 Première exécution de Frama-C

On a le warning suivant :

```
[eva] cert_exp_33.c:19:  
Frama_C_show_each_is_negative: [-2147483648..2147483647], {-1; 1}
```

En effet, dans le cas où `number` vaut `0`, `sign` n'est pas initialisé.  
On remarque l'absence de Red Alarm.

### 1.2 Afficher des valeurs en cours d'analyse

```
[eva] cert_exp_33.c:13:  
Frama_C_show_each_set_flag: [-2147483648..2147483647], {-1; 1}  
[eva] cert_exp_33.c:19:  
Frama_C_show_each_is_negative: [-2147483648..2147483647], {-1; 1}
```

### 1.3 Partitionner les traces

Avec `-eva-partition-history 1` :

```
Frama_C_show_each_set_flag: [-2147483648..2147483647], {-1; 1}}
```

Avec `-eva-partition-history 2` (ou plus...) :

```
Frama_C_show_each_set_flag: {0}, ☒  
Frama_C_show_each_set_flag: [-2147483648..-1], {-1}  
Frama_C_show_each_set_flag: [1..2147483647], {1}
```

### 1.4 Partitionnement inter-procédural

Cette fois-ci, l'état n'est pas fusionné donc on peut suivre les valeurs de `number` et `sign` dans `is_negative`.

```
Frama_C_show_each_is_negative: {0}, ☒  
Frama_C_show_each_is_negative: [-2147483648..-1], {-1}  
Frama_C_show_each_is_negative: [1..2147483647], {1}
```

On constate que cette erreur est maintenant en *Red Alarm*

### 1.5 Analyse du code corrigé

On constate l'absence d'*alarm*

## 2 CERT C EXP33-C bis

### 2.1 Première analyse

On a les erreurs suivantes :

```
[eva:alarm] cert_exp_33_realloc.c:12: Warning:  
accessing left-value that contains escaping addresses.  
assert ¬\dangling(&array);
```

```
[eva:alarm] cert_exp_33_realloc.c:38: Warning:  
accessing uninitialized left-value. assert \initialized(array + i_0);  
[eva] using specification for function printf_va_1
```

C'est la deuxième qui nous intéresse, la première est un faux positif.

### 2.2 Augmenter la précision des boucles

En précisant avec `-eva-precision 11`, on arrive à faire apparaître une *Red Alarm*

### 2.3 Analyse précise de `realloc`

Après avoir rajouté l'ACSL, on peut lancer la commande :

```
frama-c-gui -eva -eva-min-loop-unroll 10 -main func cert_exp_33_realloc.c
```

Et constater que le faux positif disparaît.

### 2.4 Analyse de la version corrigée

En lançant la version corrigée, on constate l'absence d'alertes

## 3 CWE 20

### 3.1 Première analyse

On lance une première analyse de manière « agressive » avec `frama-c-gui -eva -eva-precision 11 -main main cwe20.c`

On obtient 8 alarmes dont 3 « Red alarms »

On constate que pas mal d'alarmes sont des overflow. En effet, la validation s'assure uniquement du fait que la largeur et la longueur soit bien des entiers, mais puisqu'on les manipule après, il faut aussi s'assurer que les valeurs manipulées après restent valides.

### 3.2 Patch 1

Déjà, pourquoi utiliser un `int` au lieu d'un `unsigned int`? On représente un tableau qui ne peut pas avoir de valeurs négatives donc. Après correction et implémentation des vérifications nécessaires, on n'a pas d'erreurs d'overflow.

### 3.3 Patch 2

On vérifie ensuite que la mémoire est correctement allouée par `malloc`, ce qui nous enlève une erreur. Cependant, même après cette modification, on garde quand-même trois erreurs. On comprend en lisant la partie suivante que ce sont bien des faux positifs.

### 3.4 Améliorer la précision

### 3.5